

## Static Modifier

1. Apakah yang disebut dengan *static* variabel? Dan apa fungsi dari *static* variabel serta kapan kita dapat menggunakan *static* variabel?

Static variabel atau juga bisa disebut class variabel merupakan variabel yang menggunakan keyword *static* pada penamaannya. Dengan menambahkan keyword *static*, ini berarti variabel tidak dimiliki oleh *instance* (objek) individual dari kelas tersebut, melainkan oleh kelas itu sendiri. Akibatnya, semua *instance* dari kelas tersebut berbagi satu salinan variabel statis yang sama.

Contoh.

```
1 public class Nilai {
2     static int nilai = 0;
3
4     public void ubahNilai(int nilaiBaru) {
5         nilai = nilaiBaru;
6     }
7
8     public int bacaNilai() {
9         return nilai;
10    }
11 }
```

Pada kode di atas, kita menginisialisasi value variabel **nilai** dengan 0. Artinya, Setiap instance yang memanggil variabel **nilai** akan menghasilkan nilai "0". Sana halnya jika kita mengubah variabel **nilai**, maka seterusnya baris kode yang memanggil variabel **nilai**, value-nya akan berubah sesuai dengan yang kita tentukan. Contoh:

```
1 public class Nilai {
2     static int nilai = 0; // Variabel statis
3
4     public static void main(String[] args) {
5         Contoh obj = new Contoh();
6
7         // Mengakses dan menampilkan nilai variabel statis melalui objek
8         System.out.println("Nilai awal: " + obj.nilai); // Output: Nilai awal: 0
9
10        // Mengubah nilai variabel statis melalui objek
11        obj.nilai = 5;
12
13        // Menampilkan nilai variabel statis setelah perubahan
14        System.out.println("Nilai setelah perubahan: " + obj.nilai); // Output: Nilai setelah perubahan: 5
15    }
16 }
```

Ini berarti, setelah baris ke-11, selama belum ada perubahan value dari variabel **nilai** lebih lanjut, value-nya akan terus bernilai 5. Lalu,

kapan kita bisa menggunakan variabel *static*?

Variabel *static* dalam Java digunakan ketika kita ingin sebuah variabel dimiliki bersama oleh semua objek yang dibuat dari suatu kelas. Artinya, semua *instance* dari *class* tersebut akan mengakses dan berbagi nilai yang sama dari variabel tersebut. Ini berbeda dengan variabel *non-static* (atau variabel instance) yang memiliki salinan terpisah untuk setiap objek.

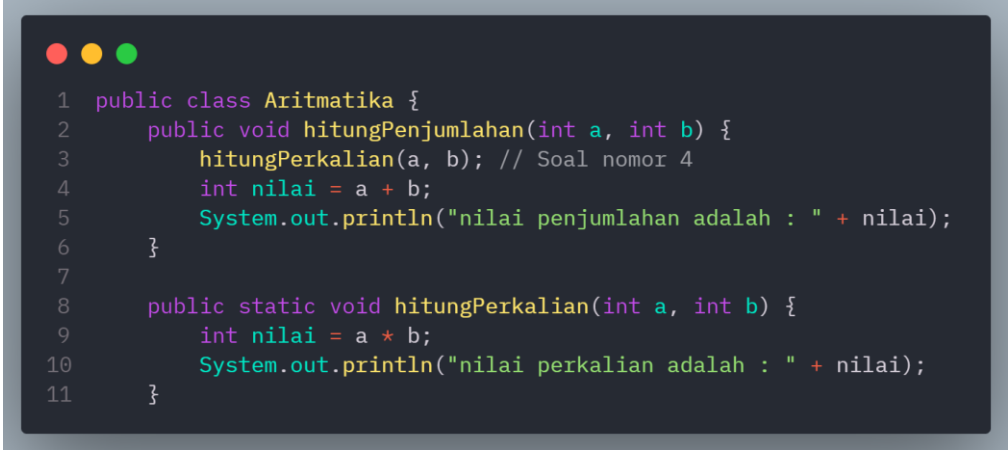
2. Mengapa pada main method harus dituliskan static? Jelaskan jawaban anda beserta dengan alasan!

Method `main()` pada Java adalah titik awal eksekusi dari sebuah program. Dengan mendeklarasikannya sebagai *static*, Java Virtual Machine (JVM) dapat memanggil metode ini langsung melalui nama *class* tanpa perlu membuat objek dari *class* tersebut.

3. Lakukan percobaan diatas dan benahi jika menemukan kesalahan!

Kode berjalan dengan baik tanpa adanya kesalahan baik itu kesalahan sintaks maupun logika.

4. Jika pada tubuh method `hitungPenjumlahan` ditambahkan syntax `hitungPerkalian(a,b)` apa yang terjadi? Jelaskan?



```
1 public class Aritmatika {
2     public void hitungPenjumlahan(int a, int b) {
3         hitungPerkalian(a, b); // Soal nomor 4
4         int nilai = a + b;
5         System.out.println("nilai penjumlahan adalah : " + nilai);
6     }
7
8     public static void hitungPerkalian(int a, int b) {
9         int nilai = a * b;
10        System.out.println("nilai perkalian adalah : " + nilai);
11    }
```

Karena `hitungPerkalian()` adalah method *static*, kita bisa langsung memanggilnya dan tidak akan ada masalah pada programnya.

Di *class* `MainAritmatika.java`, ketika method `hitungPenjumlahan()` dipanggil, method `hitungPerkalian()` pada baris ke-3 akan dipanggil dan dijalankan terlebih dahulu, maka nanti urutan eksekusinya adalah:

1. `hitungPerkalian()` dipanggil dan dijalankan
2. Menjumlahkan nilai a dan b
3. Mencetak hasil `hitungPerkalian()` serta penjumlahan a dan b

5. Jika pada tubuh method `hitungPerkalian` ditambahkan syntax `hitungPenjumlahan(a, b)` apa yang terjadi? Jelaskan?

```

1 public class Aritmatika {
2     public void hitungPenjumlahan(int a, int b) {
3         hitungPerkalian(a, b); // Soal nomor 4
4         int nilai = a + b;
5         System.out.println("nilai penjumlahan adalah : " + nilai);
6     }
7
8     public static void hitungPerkalian(int a, int b) {
9         hitungPenjumlahan(a, b); // Soal nomor 5 [error karena method ini static]
10        int nilai = a * b;
11        System.out.println("nilai perkalian adalah : " + nilai);
12    }

```

Di sini baris ke-9 akan mengalami error. Jika kita mengarahkan cursor ke bagian yang error, maka akan terlihat pesan error-nya yaitu:

***Cannot make a static reference to the non-static method hitungPenjumlahan(int, int) from the type AritmatikaJava(603979977)***

Intinya adalah kita tidak bisa memanggil method *non-static* di dalam method *static*. Salah satu penyebabnya adalah, method dengan keyword *static* berlaku untuk semua instance di class Aritmatika. Method *static* tidak memiliki akses ke data atau method yang spesifik untuk suatu instance. Oleh karena itu, method *static* tidak dapat secara langsung memanggil method *non-static* karena method *non-static* memerlukan konteks instance tertentu untuk dijalankan.

6. Tambahkan method non static dengan nilai balikan double untuk menghitung pembagian dengan parameter String nil dan String nil2, dan panggil method tersebut pada method main!

```

1 // Soal nomor 6
2 System.out.print("masukkan nilai 1: ");
3 String strNil1 = in.next(); // Menggunakan String
4 System.out.print("masukkan nilai 2: ");
5 String strNil2 = in.next(); // Menggunakan String
6 double hasilPembagian = a.hitungPembagian(strNil1, strNil2);
7 System.out.println("nilai pembagian adalah : " + hasilPembagian);

```

```

1 // Soal nomor 6
2 public double hitungPembagian(String nil, String nil2) {
3     double nilai = Double.parseDouble(nil) / Double.parseDouble(nil2);
4     return nilai;
5 }

```